PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-288648
(43)Date of publication of application : 04.11.1997

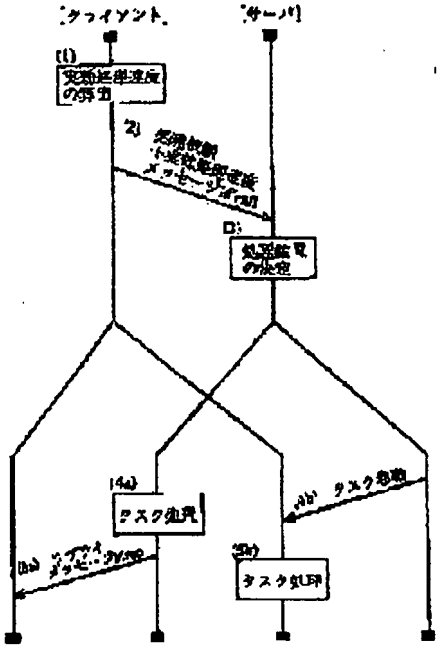
(51)Int.Cl. G06F 15/16
G06F 13/00

(21)Application number : 08-098174
(22)Date of filing : 19.04.1996

(71)Applicant : MATSUSHITA ELECTRIC IND CO LTD
(72)Inventor : WAKAYA AKIYOSHI

(54) DISTRIBUTED PROCESSING METHOD IN NETWORK AND ITS SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To provide an efficient distributed processing system coping with the fluctuation of load conditions in a network.
SOLUTION: A client calculates its own effective processing speed (procedure 1) and requests the processing of a task provided in a server to the server along with the information (procedure 2). The server judges which device is to process the task so as to shorten turn-around time relating to the task based on the received processing speed of the client and its own processing speed (procedure 3). By the judgement, the server himself processes the task (procedures 4a and 5a) or the client processes it (procedure 5b) after moving the task to the client (procedure 4b).



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平6－243112

(43)公開日　平成６年(1994)９月２日

(51)Int.Cl.[5]		識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ	15/16	３８０　Ｚ	7429－5L		
	9/46	３６０　Ｂ	8120－5B		
	13/00	３５５	7368－5B		

審査請求　未請求　請求項の数２　ＯＬ　（全 10 頁）

(21)出願番号　特願平5－30613

(22)出願日　平成５年(1993)２月19日

(71)出願人　000002369
セイコーエプソン株式会社
東京都新宿区西新宿２丁目４番１号

(72)発明者　長坂　文夫
長野県諏訪市大和３丁目３番５号　セイコーエプソン株式会社内

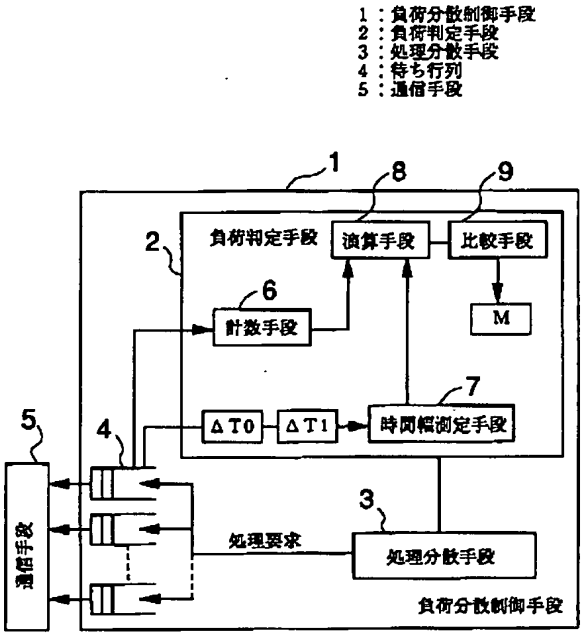
(74)代理人　弁理士　鈴木　喜三郎　（外１名）

(54)【発明の名称】　マルチプロセッサ装置

(57)【要約】

【目的】異機種混在のネットワーク環境でのマルチプロセッサ並列処理システムにおいて、他プロセッサの負荷検出を行ない、最適負荷分散を図る。

【構成】他プロセッサに処理単位を分散し、並列実行によって処理時間の短縮を図る際にプロセッサ毎に待ち行列４および負荷判定手段２を設ける。負荷判定手段２は待ち行列４の長さ、及び前回までの処理時間の実績ΔＴ0、ΔＴ1を用いて演算を行ない、次の処理の終了時間の期待値Ｍを返す。処理分散手段３はこのＭの値の最も小さいプロセッサ装置に処理を分散し、待ち行列４にエンキューする。また処理分散手段３はどれか処理が完了となった場合、処理時間を記録すると共に待ち行列４から処理単位をデキューする。負荷分散手段１は負荷判定手段２、処理分散手段３、待ち行列４によって構成され、通信を用いることなく他プロセッサの負荷を知り、処理分散を行なう事が出来る。



1

【特許請求の範囲】

【請求項1】　ローカルエリアネットワークによって接続された複数台のワークステーションあるいはパーソナルコンピュータなど（以下、総称してプロセッサ装置と書く）を用いて、一つの目的プログラムの処理を複数のプロセスに分割し、これを複数のプロセッサ装置に分散して並列して実行することを特徴とするマルチプロセッサ装置において、並列処理開始を記述した文を含むプログラムの一単位（これを並列処理要求の発生源プロセスと書く）が、該プロセス中の分散記述に従って発生した子プロセスを複数のプロセッサ装置に配置する配置手段と、前記の複数のプロセッサ装置に子プロセスを分散処理要求するための負荷分散制御手段と、を有する事を特徴とするマルチプロセッサ装置。

【請求項2】　前述の負荷分散制御手段が、処理分散対象のプロセッサ装置ごとに要求プロセスの待ち行列を作成する待ち行列作成手段と、前記要求プロセスの発生から完了までの処理時間を計測する計数手段と、前記待ち行列作成手段によって作成された要求プロセスの待ち行列の長さと、前記計数手段によって計測された前回以前の要求プロセスの処理時間と、をパラメータとする演算によって個々のプロセッサ装置の負荷を判定する負荷判定手段と、該負荷判定手段の結果に従って処理を分散する処理分散手段と、を有することを特徴とする請求項1記載のマルチプロセッサ装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、ローカルエリアネットワークに代表される通信手段を用いて接続した複数のプロセッサにより構成されるマルチプロセッサシステムにおいて、目的プログラムの並列実行を行う際に負荷を均等に分散させる手法に関する。

【0002】

【従来の技術】マルチプロセッサシステムによって処理対象のプログラムを並列処理し、処理の高速化を図る技術では、各プロセッサの負荷が均等になる様に処理単位を配置する手段が重要である。

【0003】個々のプロセッサに過度の負荷の集中あるいは処理の過疎が発生することのないように処理単位を配置するためには、各プロセッサの負荷を検出する機構が必要である。この機構の実現には、プロセッサの結合状態について、共有メモリの有無、通信速度の高低などを考慮した手段が要求される。

【0004】ローカルエリアネットワーク等の通信手段を用いたプロセッサ間の疎結合の並列処理では、プロセッサ負荷検出のための通信時間が処理時間の多くの比率を占めてはならない。言うまでも無く、このような本来の処理目的に不要な処理は可能な限り排除し、システム全体のプロセッサ資源を目的処理に集中することが重要である。

2

【0005】この分野に対してなされた従来の発明には、特開昭63-211060号がある。この従来発明は、「自プロセッサの負荷状態を、実行プロセス数や待ち行列の長さ等から常時判断し、高負荷に変化した時、または低負荷に変動した時、通信手段により他の全プロセッサに負荷状態を通知する」という処理を特徴としている。また、この通信により得られた他プロセッサの負荷状態と自プロセッサの負荷状態は個々に記録され、プロセスの処理分散の時点で参照される構成である。

【0006】この従来発明は技術要素に分解すると次の4点で構成された発明と言える。

【0007】（1）個々のプロセッサは自分の負荷状況を常時監視する。

（2）個々のプロセッサはあらかじめ負荷の値の閾値を定め、これを上回った時点と下がった時点で他の全プロセッサへ通知する。

（3）個々のプロセッサは他プロセッサ、自プロセッサの負荷の最新の値を記録として保持している。

（4）プロセッサの負荷は処理中のプロセス数と待ち行列の長さによって判定される。

【0008】

【発明が解決しようとする課題】しかし、従来発明は上記（1）と（2）の不必要な処理によってプロセッサ資源およびネットワーク資源を消費するという問題点があった。（1）と（2）が不要であるのは以下の2つの理由による。

・あるプロセッサの負荷はこのプロセッサに処理分散をしようとするプロセスが調べれば良く、要求が無いにも係らず常に自プロセッサの負荷を調べる必要は無い。

・各プロセッサが負荷変動のある都度にネットワーク上の全プロセッサに通信（ブロードキャスト）を発生する事は、不必要な通信でネットワークの負荷の増大を招く。

【0009】結局、複数のプロセッサ間の疎結合の並列処理において、最適負荷分散のために必要な構成技術は、処理分散対象のプロセッサ（自プロセッサも含め）の負荷を知る手段だけである。

【0010】さらに、従来発明にはもう一つの問題点があった。それは、各プロセッサ装置の性能の差で生じる処理時間の不均質を評価出来ない点である。これは上記（4）の構成から生じる問題点である。

【0011】この問題は、ワークステーション等の性能は著しく向上し、1年で2倍程度の処理速度向上が達成されているため、一つのネットワーク上に数倍～10数倍の処理速度差のあるプロセッサ装置群が接続される異機種混在環境が現れたために生じたものである。つまり、異機種混在環境における並列処理では各プロセッサの処理待ち行列の長さが同じであってもプロセッサの負荷は同じとは言えないために、次の処理要求の完了時刻の期待値が等しくはならないということである。

3

【0012】

【課題を解決するための手段】この様な課題を解決するために本発明のマルチプロセッサ装置では、ローカルエリアネットワークによって接続された複数台のプロセッサ装置を用いて一つの目的プログラムの処理を複数のプロセスに分割し、これを複数のプロセッサ装置に分散して並列して実行する場合において、並列処理開始を記述した文を含む並列処理要求の発生源プロセスが、該プロセス中の分散記述に従って発生した子プロセスを複数のプロセッサ装置に配置する配置手段と、前記の複数のプロセッサ装置に子プロセスを分散処理要求するための負荷分散制御手段と、を有し、しかも前述の負荷分散制御手段が、処理分散対象のプロセッサ装置ごとに要求プロセスの待ち行列を作成する待ち行列作成手段と、前記要求プロセスの発生から完了までの処理時間を計測する計数手段と、前記待ち行列作成手段によって作成された要求プロセスの待ち行列の長さと前記計数手段によって計測された前回以前の要求プロセスの処理時間とをパラメータとする演算によって個々のプロセッサ装置の負荷を判定する負荷判定手段と、該負荷判定手段の結果に従って処理を分散する処理分散手段と、を有することを特徴としている。

【0013】

【作用】本発明のマルチプロセッサ装置は、異機種混在環境で処理の多重化（並列化）による処理時間短縮の効果を最大限に利用するために、次の処理要求完了時間が均等になる様に処理単位を分散するよう作用する。

【0014】

【実施例】本発明の実施例について以下の項目に従い、図を用いて説明を行なう。

【0015】１．本実施例の特徴（従来発明との違い）

２．負荷検出手段の動作

２．１　オペレーティングシステムによるマルチタスク処理の概要

２．２　処理要求発生源プロセス２２の動作

２．３　負荷分散制御手段１の動作

３．説明の補足（並列処理の実現）

１．本実施例の特徴（従来発明との違い）

本発明が実施されるのに好適な一つの環境の説明図を図３に示す。本実施例は、ワークステーション、パーソナルコンピュータなど、プロセッサ資源を有する複数台の情報機器がネットワーク２１によって接続された環境での、プログラムの並列処理を主な応用の対象としている。

【0016】目的プログラム２２はパーソナルコンピュータ３３で起動される。２２に含まれる並列処理のオブジェクトコードが複数のプロセス３６、２０６らを発生する。これらプロセス３６、２０６は、ネットワーク２１を介して処理分散される。パーソナルコンピュータ３４あるいはワークステーション２００は、プロセス３６、２０６らを実行する。目的プログラム２２は、このプロセス３６、２０６が並列して実行される事によって逐次実行に比較し短時間で処理を終える事が出来る。

4

【0017】ここでプロセスとは、オペレーティングシステムにおける、プロセッサ資源、メモリ資源割り当ての実行時の単位である。本実施例のプロセスの構造を図４に示す。本実施例において、プロセス４０は、プロセスの識別子４１及び実行管理、メモリ管理のための情報を含むプロセス管理情報４２と、中断の際に現在のプロセッサのレジスタの状態を保存するためのカーネルスタック４３と、オブジェクト領域４６、プロセススタック４４、ヒープ領域４５からなる管理単位であるとして説明を続ける。この意味で目的プログラム２２は、やはり一つのプロセスと見なせる。以下の説明では、”最初に処理並列化を要求する”という意味で目的プログラムを「処理要求の発生源プロセス」と呼ぶ。

【0018】本実施例のマルチプロセッサ装置と、従来発明（特開昭63-211060号）との大きな違いは次の３点である。

【0019】（１）本実施例の処理要求の発生源プロセスは、他のプロセッサへの負荷の均等な分散をそれぞれ管理する。

（２）本実施例の処理要求の発生源プロセスは、他プロセッサの負荷判定に際し、他のプロセッサの状態を通信により知る処理を必要としない。

（３）本実施例の処理要求の発生源プロセスは、他のプロセッサの負荷を判定する際、待ち行列の長さ情報に加え、前回までの処理時間の情報を使用する。

【0020】以上の（１）～（３）によって本実施例では発生源プロセスが負荷の判定と処理分散の動作を完全に自己の処理で完結する。これを「自律的な判定」と書く。すなわち本実施例が従来発明に比較し異なる最大の点は、他のプロセッサ装置の負荷を自律的に判定する点である。これによって、従来発明で行なわれた負荷変動通知のためのデータ転送は不要になる。

【0021】以上に挙げた本実施例の特徴を次に詳しく説明する。

【0022】２．負荷検出手段の動作

図２は本実施例のマルチプロセッサ装置全体の概略構成図である。また図１は、本実施例の中において特徴的な構成を実現する、負荷分散制御手段１の構成図である。

【0023】本実施例の特徴をなす負荷分散制御手段１は、負荷判定手段２、処理分散手段３、待ち行列４から構成される。負荷分散制御手段１は、本実施例のマルチプロセッサ装置の処理分散を管理する機構であり、単独のワークステーション２００等に実装されたオペレーティングシステム２０１のスケジューラ２０２とは異なる。この点を明らかにするために、図２によって全体の動作を説明する。

【0024】２．１　オペレーティングシステムによる

5

マルチタスク処理の概要

複数台のワークステーション２００が、ネットワーク２１を介し接続されている。あるパーソナルコンピュータ３３で実行されている処理要求発生源プロセス２２は、説明の簡単のために、直接ネットワーク２１に接続されているように図示した。ワークステーション２００は、オペレーティングシステム２０１（以下ＯＳ２０１と略）によって管理される。ＯＳ２０１はマルチタスクを行なう。このためにＯＳ２０１は、スケジューラ２０２と、これに対する待ち行列２０５を持つ。ＯＳ２０１は、複数のユーザプログラムの中からどれか一つのユーザプログラムを選んで実行する。ある条件が成立すると（後述）、ＯＳ２０１はこのユーザプログラムの実行を一時中断し、他のユーザプログラムを実行する。これによってＯＳ２０１はマルチタスクを実現する。

【００２５】一つのユーザプログラムは、ＯＳ２０１の管理下で一つのプロセスとして起動される。起動後にこのプロセスがＯＳ２０１の機能のサービスを受けて、更に別のプロセスを作る事が許される。前述したユーザプログラムのマルチタスクは、ＯＳ２０１の管理から見るとユーザプログラムを形成する一つないし複数のプロセスに対する「プロセッサ割り当ての切り替え」によって実現される。

【００２６】あるプロセスがＯＳ２０１の機能のサービスを利用する手段は、ソフトウェア割り込みである。この割り込みが発生すると、プロセッサのプログラムカウンタがＯＳ２０１内部の処理番地の値に書き変わる。これによりプロセッサ処理は、ユーザのプロセスからＯＳ２０１の内部に入る事ができる。

【００２７】ユーザプロセスからの処理要求（上記ソフトウェア割り込み）が発生すると、プロセッサ処理は最初にアプリケーションインターフェース２０４に移る。ここで要求の処理内容が判断され、サービスプログラム２０３が実行される。一部の要求（ハードディスク装置からの読み込み等）は、実際の処理完了までに時間待ちを伴う。この場合、要求を発生したプロセスは休眠状態に入る。すなわち、プロセスが待ち行列２０５にエンキューされ、次にＯＳ２０１によって起動されプロセッサ割り当てを受けるまで停止する。

【００２８】また、これ以外の場合でもプロセスが予め定められた数ミリ秒～数１０ミリ秒のプロセッサ割り当て時間を消費した場合は、タイマ割り込みによってＯＳ２０１の内部の処理プログラムが動作し、このプロセスを停止する。この場合も、プロセスが待ち行列２０５にエンキューされ、次にＯＳ２０１によって起動されプロセッサ割り当てを受けるまで停止する。

【００２９】ＯＳ２０１はタイムスライスの経過または処理待ちを伴うサービスで、プロセスを休眠させるとスケジューラ２０２を用いて次に実行するプロセスを選択する。以上がＯＳ２０１によるマルチタスク処理の概要である。

6

【００３０】２．２　処理要求発生源プロセス２２の動作

本実施例の処理要求発生源プロセス２２は、ＯＳ２０１と同等のオペレーティングシステムに管理される一つのプロセスである。処理要求発生源プロセス２２はユーザプログラムとして起動され、cobegin文等（３節で後述する）に代表される並列記述により処理単位Ｊ１、Ｊ２、Ｊ３を並列して動作させる。ここで、処理単位のどれか、例えばＪ２内にさらに並列処理記述がふくまれる場合、次はＪ２が新たな処理要求発生源プロセス２２として動作する。

【００３１】処理要求発生源プロセス２２の目的コードは、ソースコードをコンパイルして得られたものであり、特定のプロセッサの機械語で構成される。従って、プロセス２２のオブジェクトコードをそのまま他のワークステーション２００に転送しても動作は保証されない。本実施例は、異機種コンピュータ間の並列処理を目的とするので、この不都合を除くために仮想プロセッサとその機械語を導入する。仮想プロセッサの機械語は特定のハードウェアに依存しないものであり、仮想機械語のインタープリタ２１２によって実行される。

【００３２】処理要求発生源プロセス２２と処理受け取りプロセス２０７は、そのプロセスの構造上は同じであり、プロセスの構造４０（図４）からなる。オブジェクト領域４６は各プロセスの動作を記述したオブジェクトコードを格納した領域であり、仮想機械語のインタープリタ２１２はこの領域に含まれる。また、仮想機械のプログラムカウンタ、スタックポインタ、レジスタ、フラグから構成される仮想プロセッサ２１１は、各プロセスのヒープ領域４５に作られる。また、プロセス自体が動作のためにメモリ領域を必要とするが、この領域はプロセス内に作業メモリ２１０として確保される。

【００３３】仮想機械によるプロセス４７はヒープ領域４５に配置される。プロセス４７は、仮想機械の機械語で形成されるオブジェクトコードと、作業メモリ（ヒープ、スタック）からなるデータ構造で、ＯＳ２０１が管理するプロセスの構造４０を仮想機械でシミュレートした構造である。

【００３４】以下の説明において、処理の流れは図５に示した。

【００３５】プロセス２２の処理単位Ｊ１、Ｊ２、Ｊ３はいずれも仮想機械によるプロセス４７である（説明のため、同様の処理単位を一般にＪｎと書く）。プロセス２２は、cobegin文による分岐２３によってランタイム・ルーチン２４を呼び出してＪ１、Ｊ２、Ｊ３をどのプロセッサ装置に分散すべきか判断し、処理分散を行なう。

【００３６】cobegin文による機械語は、並列処理モードであることを内部フラグに記録する処理を行ない（Ｓ

５０１）、実際の処理分散は処理単位Ｊｎを呼び出す機械語のランタイム・ルーチン２４が行なう。

【００３７】負荷判定手段２は、各プロセッサ装置毎に用意した待ち行列４から、処理完了待ちのプロセス数（待ち行列の長さ）と、前回までの処理時間とを取り出し、プロセッサ装置の負荷を求め、次の処理の完了時間の期待値Ｍを返す（Ｓ５０２）。処理分散手段３は、この値Ｍに基づき、最も処理完了時間の予測値の短いプロセッサ装置の装置番号Ｐを返す。すなわち、負荷判定手段２の返したＭが現在最小の値より小さければ（Ｓ５０３）、最小値の更新およびプロセッサ装置の装置番号Ｐの記録を行ない（Ｓ５０４）、これを全プロセッサ検査完了まで繰り返す（Ｓ５０５）。

【００３８】負荷分散制御手段１は、上記手順で見つけた装置番号Ｐのプロセッサに処理単位Ｊ１を転送し、処理要求を発行する（Ｓ５０６）。Ｊ２、Ｊ３についても同様の手段が行なわれる。

【００３９】もちろん処理の一部は自プロセッサ装置で実行されても良い。待ち行列４、および負荷判定手段２は自プロセッサ装置に対しても作られるので、負荷分散制御手段１が、自プロセッサ装置に処理単位Ｊｎを割り当てる事がある。この場合Ｊｎを形成する仮想機械のプロセス４７は、自プロセッサ上に作られた新しいプロセス２０７で処理される。割り当ては、とにかく終了時間の期待値が最も早いプロセッサ装置を指定するだけである。

【００４０】上記から分かる様に、処理単位Ｊｎは、他プロセッサ装置に送られる場合とそうでない場合がある。通信手段５は、処理単位Ｊｎと、装置番号Ｐを受け取り、処理の転送を行なう。通信手段５は、受け取った装置番号Ｐが自プロセッサの値であれば、通信せず処理し、それ以外の場合、通信を発生し、他プロセッサ装置の処理受け取りプロセス２０７と接続する（Ｓ５０７）。

【００４１】プロセス２０７は、ネットワーク２１を介して送られてくる処理単位Ｊｎを処理し、処理完了時点で処理完了を通知し、処理結果を返す動作をする。既に述べた様に、Ｊｎは仮想機械のプロセス４７の構造をとるから、処理結果はプロセス４７のスタックに残される。

【００４２】プロセス２０７は、ネットワーク２１に接続された複数のプロセッサ装置らからランダムに到着する要求に応えなければならない。そのために、処理単位Ｊｎを受け取った時プロセス２０７は直接これを実行することはせず、子プロセス２０６を生成し、処理を行なう。プロセス２０７は、通信手段２０８によって処理単位Ｊｎを受け取ると、プロセス生成・起動処理２０９を実行する（Ｓ５０８）。プロセス生成・起動処理２０９は、ＯＳ２０１のアプリケーションインターフェース２０４を利用し、ＯＳ２０１に子プロセス２０６の生成を要求する（Ｓ５０９）。ＯＳ２０１はサービスプログラム２０３を実行し、子プロセス２０６を作る。

【００４３】ＯＳ２０１の子プロセス生成は、「親のプロセスと同じものを複製する」処理である。この時、スタックの内容、プログラムカウンタ等各種レジスタの値、全てが親プロセスの構造４０と同一に作られる。従って、子プロセス２０６での次の実行開始番地は、直前に親プロセスで実行されていた番地に引き続く番地である。親プロセス２０７では、この直前に仮想機械語のインタープリタ２１２を実行する処理位置であった。より詳しく言うと、通信手段２０８から仮想機械語を受け取り、インタープリタ２１２に入力として与える位置であった。一方、受け取った仮想機械語のデータ列は、仮想機械のプロセス４７の構造をとっており、このプロセス４７のプログラムカウンタは、cobegin文から生成された仮想機械語に続く、次の命令語の位置を指し示している。そこで、子プロセス２０６も仮想機械語のインタープリタ２１２処理に入り、cobegin文の次に続く文の位置から処理を開始し、Ｊｎを処理する事ができる（Ｓ５１０）。

【００４４】プロセス２０７はプロセス２２と同一の構造である。プロセス２０７が受け取った処理単位Ｊｎが更に並列処理を含んでいる場合、今度はプロセス２０７が作った子プロセス２０６が新しい並列処理要求の発生源のプロセスとなる。

【００４５】以上で処理分散に関わる処理要求発生源プロセス２２の動作の説明を終わる。続いて、負荷分散制御手段１の動作を説明する。

【００４６】２．３　負荷分散制御手段１の動作

負荷分散制御手段１の構成は図１で示される。既に述べた様に、負荷判定手段２、処理分散手段３、待ち行列４が主たる構成要素である。「１．本実施例の特徴」に述べた様に、本実施例では通信を発生して他のプロセッサ装置の負荷を調べる必要が無い。その代わり自プロセッサも含め、利用出来る全てのプロセッサ装置について待ち行列４と、負荷判定手段２を用意する図１は、分かりやすくするために負荷判定手段２を一つだけ取り出して記載した。

【００４７】利用出来る他プロセッサ装置の装置番号は予め自プロセッサ装置に初期設定として与える。また各プロセッサ装置はマルチタスク処理出来ることが必要条件であり、予め処理受け取りプロセス２０７がバックグランドで動作している事が必要である。

【００４８】既に「２．１　…マルチタスク処理の概要」に書いた様に、各プロセッサ装置２００において、ＯＳ２０１がプロセス２０７に処理時間を割り当て、並列処理の要求が実行される。プロセス２２が処理単位Ｊｎを分散してから処理結果が得られるまでには、時間の遅延が発生する。そこで処理分散手段３はＪｎをどれかのプロセッサ装置２００に送った時点でその処理単位Ｊ

9

n（仮想機械によるプロセス47と見なされる）にプロセス番号を与え、待ち行列4に記録する（エンキュー）。同じく処理分散手段3はJnが処理完了し、処理結果が戻った場合は、その時点で待ち行列4から該当するプロセス番号のエントリを取り除く（デキュー）。

【0049】処理分散手段3は、エンキューからデキューまでの時間を待ち行列4が併せ持つデータ構造に記録する。この処理の結果、負荷判定手段2は待ち行列4のデータ構造を調べ、現在の待ち行列4の長さと、前回までの処理時間の実績値を取り出すことが出来る。計数手段6は待ち行列4の長さを取り出す。時間幅測定手段7は前回までの処理時間の実績値を取り出す。

【0050】メモリスペースが許せば、待ち行列4のデータ構造に前回までの全ての処理の処理時間を記録することも可能である。しかし、それによって次の処理完了時間の期待値が正確に求まる保証は無い。そこで、本実施例では直前の処理時間（これをΔT1と書く）とそれ以前までの実績値（これをΔT0と書く）を用いて今回実績時間(これをΔTと書く）を決定した。すなわち

ΔT＝（ΔT1＋ΔT0）÷2

とした。またこの次以降の実績値は、現在待ち行列4にある未完了のどれかの処理が完了した時点で更新される。すなわち新たに完了した処理のプロセス番号が待ち行列4からデキューされ、処理時間ΔT1が更新される。次に代入

ΔT0 ＝ ΔT

により、実績値が新しい値で更新される。

【0051】演算手段8は、計数手段6が取り出した待ち行列の長さと、時間幅測定手段7が求めた今回実績値ΔTの積を求める。これを

M＝（待ち行列の長さ）×ΔT

で表す。比較手段9はここで求めたMが予め定めた値を越えた時、「処理不可能」を返し、それ以外は値Mをそのまま返す。また、今回初めて処理を割り当てる場合は、実績値の値として予め定めたΔTの値を与える。また、待ち行列の長さは＝1である。

【0052】再び図2に戻り説明を続ける。負荷判定手段2は比較手段9の値により、各プロセッサ装置200の負荷を返す。処理分散手段3は、「処理不可能」な場合を除いて戻り値であるMを比較し、最も短い（つまり早く処理完了すると期待出来る）プロセッサ装置を選択し、処理要求を発行する。

【0053】処理単位Jnの大きさ、負荷は様々であり、処理時間には大きなバラツキがあると予想される。従って、本実施例の方法が正しく処理時間の期待値を与える事にはならない。しかし、処理要求自体がランダムに発生すること、処理単位Jnのバラツキもランダムであることを考えると、およそ適切な負荷配分を定める方法であると言える。しかも、負荷検出に通信を使用しない点が大きな特徴である。

10

【0054】これまでの説明は、負荷分散制御手段1がプロセス22の中に実現されている場合について書いた。しかし、以上の説明から明らかな様に負荷分散制御手段1は、プロセス22の中で処理の並列化要求が有った場合、プロセッサ装置（の装置番号）を指定して処理を分散する機能を実現出来れば良い。そこで別の実施例として負荷分散制御手段1だけを独立したプロセスとする方法が考えられる。この場合は、負荷分散制御手段1のために設けたプロセスに対し、並列処理要求の発生源プロセス22が、プロセス間通信によって処理依頼を発行すれば良い。負荷分散制御手段1の独立プロセスは常時バックグランドに走行させる事が可能で有る。実際、多くのオペレーティングシステムにバックグランド処理のための機能と、プロセス間通信の機能が用意されている。従って、負荷分散制御手段1を独立のプロセスとする事は簡単な変更である。この場合も処理内容は上記手順と同様である。

【0055】また負荷判定手段2も別の方法が考えられる。一例として前回までの実績値を指数平滑法などで処理し、次回の処理時間予測値を求めれば良い。いずれにせよ本実施例は、過去の処理時間の値と待ち行列の長さに基づき、他プロセッサの負荷判定を行なう事が特徴の一つであり、これによって異機種混在環境のマルチプロセッサ装置の最適負荷分散を行なうものである。

【0056】以上で本実施例における負荷判定と処理分散の手段の説明を終わる。次に並列記述の手段について説明の補足をする。

【0057】3．説明の補足（並列処理の実現）

並列処理を行なうために何らかの並列モデルに従い、問題の記述と実行のできる処理系が必要である。この処理系の形式として、比較的容易に実現出来るのは次の3通りである。

【0058】（A）相互接続された複数のプロセッサ資源、メモリ資源を動的に管理できるOSを用意し、このOSの管理下で並列記述が可能な言語仕様の開発言語を用いて目的プログラムの記述、コンパイル、実行を行なう。

【0059】（B）目的プログラムに対する任意の開発言語（C,Fortran,Pascal等の汎用言語）と、これら開発言語用の「遠隔手続き呼び出し」のライブラリを用意し、個々の要求に応じて、毎回目的プログラムを書き下し、コンパイルと実行をオペレーティングシステム（以下OSと略）の管理下で行なう。

【0060】（C）並列記述が可能な言語仕様の開発言語を用意し、目的プログラムを記述し、コンパイルと実行をOSの管理下で行なう。

【0061】以上の中で（A）の実現には少なくとも同一のOSを実装したプロセッサ装置からなるマルチプロセッサ環境が必要である。現時点（1992年）でより一般的な（B）と（C）は実質的に同様の処理を行なう

ものである。（B）の場合、使用者は「遠隔手続き呼び出し」の個々の関数仕様に基づいて、正しい手順、正しい引数を満たす様にソースコードを書く事が要求される。一方使用者が（C）の方式を用いる場合は、高水準の言語記述によって実行の並列化を記述する事が可能である。よって（C）の方式が記述容易性に優れている。

【００６２】本実施例では、（C）の方式を用いて並列記述を行なった場合を例にとり説明を加えた。方式（C）に必要な事項と、これに対する本実施例の手段を次に挙げる。

【００６３】要請３．１：（B）の「遠隔手続き呼び出し」または、（A）の動的資源管理に代替する手段が必要。

【００６４】手段３．１：本実施例は、通信手段２０８、プロセス生成・起動処理２０９により構成された処理受け取りプロセス２０７を、並列処理対象の各プロセッサ装置２００で予め実行する。これは、プロセッサ装置２００の起動後直ちに実行され、バックグランドで走り続けるプログラムとして設定することによって実現される。現在多くのオペレーティングシステムでこのような処理手順は周知である。

【００６５】要請３．２：プロセッサの機械語をはじめとし、アーキテクチャの異なる機器において、プログラムの処理単位（この実施例では個々のプロセス）の目的コードを実行するための手段が必要。

【００６６】手段３．２：本実施例は、プロセッサ固有の機械語に依存しない仮想プロセッサ２１１を論理的に作りだした。実際の処理は仮想プロセッサ２１１の機械語を用い、個々のプロセッサ装置２００上で、この仮想機械語のインタープリタ２１２実行する。処理受け取りプロセス２０７は、ＯＳ２０１のサービスを利用し自己と同一のプロセスの複製２０６を生成し、他プロセッサ装置からの要求を処理する。

【００６７】要請３．３：目的プログラムを記述する段階で処理対象の問題に有る並列性を取り出し、ソースコード中に記述するための開発言語が必要。

【００６８】手段３．３：本実施例では簡単のため、「２．２　処理要求発生源プロセス２２の動作」で、"cobegin文による並列処理"と書いた。cobegin文はPascal系の言語を並列処理向けに拡張した際に用いられる事が多い。図６は言語Pascalに{cobegin, coend}の２語を拡張した場合の記述の例を示した図である。プログラムリスト６００は、Ｐ１という名前を持つプログラムが、手続きＪ１、Ｊ２、Ｊ３を持ち、これを並列に実行（６０１）するという内容の例である。

【００６９】言うまでもなく、手段３．３は目的プログラムを作成する段階で並列処理を記述することが出来る仕様であれば、どんな言語を用いても良い。リスト６００は、例えば最も簡単な場合、リスト６０２の機械語リストの様にコンパイルできる。ここで、各命令語は次の通りである。

【００７０】(1) JP　　プログラムカウンタを書き換え処理を移動する（ジャンプ）
(2) COBEGIN　cobegin文の置き換え
(3) COEND　coend文の置き換え
(4) JSR　　サブルーチンへの分岐
(5) HALT　　処理停止

リスト６０２の処理において、COBEGIN行６０３では、他プロセッサ装置との通信の確立、および並列実行モード内部フラグ設定を行なう。続くJSR行６０４は、並列処理時と、それ以外の場合で処理内容が異なる。

【００７１】（並列処理時）流れ図５に示した一連の処理を行なう。

【００７２】（逐次処理時）現在の処理番地の次の番地を戻り番地として記録した後、指定された番地へジャンプする。

【００７３】本実施例は以上の手段で並列処理を記述した。

【００７４】

【発明の効果】以上の実施例が明らかにした様に、本発明の構成要素の一つである負荷分散制御手段は、他プロセッサへの処理要求待ち行列の長さと、前回までの応答時間実績値に基づいて、対象となるプロセッサの負荷を自律的に判断する。この結果、負荷検出に伴う通信が不要となるために、ネットワーク資源、プロセッサ資源のそれぞれに関し、目的プログラムの処理に割り当てる時間の比率を向上させることが可能となる。

【００７５】さらに、負荷分散制御手段の構成要素である負荷判定手段が、処理時間の時間幅測定手段を持つ構成とすることで性能の異なる様々なプロセッサ装置からなる混在環境においても前回実績時間を加味してプロセッサの処理速度を反映した負荷判定を行なう事が可能である。この結果、結合されるプロセッサ装置の速度特性を事前に設定する等の手順を用いる事無く、自動的に最適負荷分散を行なう事ができる。これは、マルチプロセッサ装置全体の運用の容易性を向上させる効果が多大である。

【図面の簡単な説明】

【図１】本発明の特徴である負荷分散制御手段の説明図。

【図２】本発明の一つの実施例の構成図。

【図３】図２の実施例の実行環境の説明図。

【図４】実施例で用いたプロセスの構造の説明図。

【図５】実行単位分散の処理手順の流れ図。

【図６】並列処理記述の例の説明図。

【符号の説明】

１…負荷分散制御手段
２…負荷判定手段
３…処理分散手段
４…待ち行列

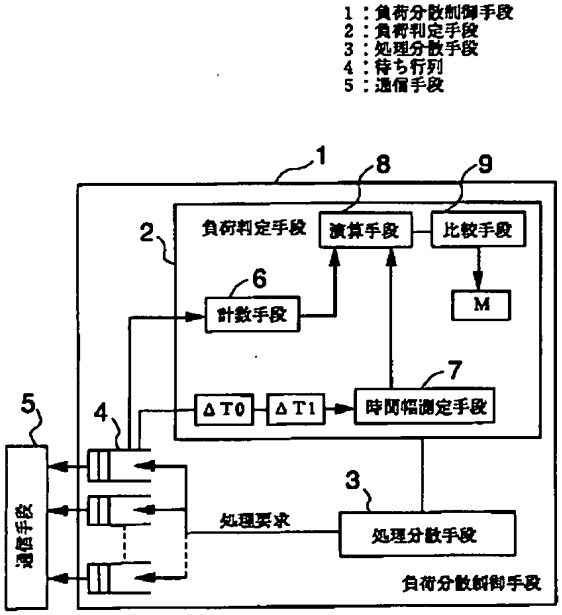
13

5…通信手段
6…計数手段
7…時間幅測定手段
8…演算手段
9…比較手段
21…ネットワーク
22…並列処理要求発生源プロセス（目的プログラム）
23…分岐
24…ランタイム・ルーチン
33、34…パーソナルコンピュータ
36…並行して実行されるプロセス
40…プロセスの構造
41…プロセス識別子
42…プロセス管理情報
43…カーネルスタック
44…プロセススタック

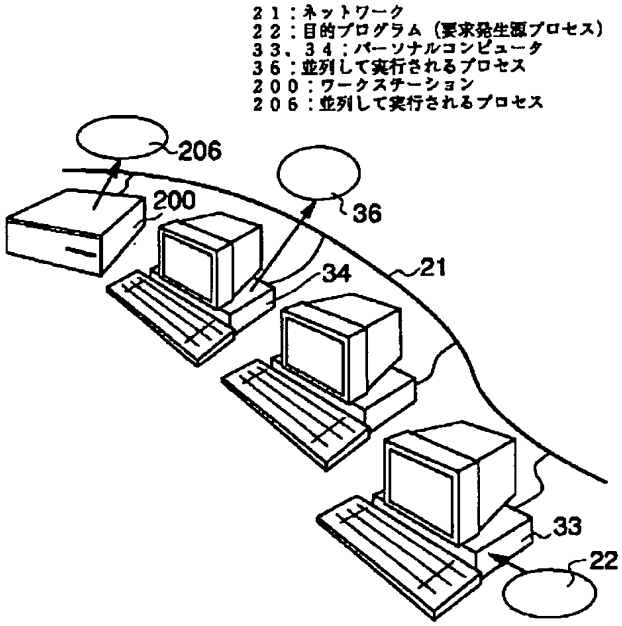
14

45…ヒープ領域
46…オブジェクト領域
47…仮想機械のプロセス
200…ワークステーション
201…オペレーティングシステム
202…スケジューラ
203…サービスプログラム
204…アプリケーションインターフェース
205…待ち行列
206…子プロセス
207…処理受け取りプロセス
208…通信手段
209…プロセス生成・起動処理
210…作業メモリ
211…仮想プロセッサ
212…仮想機械語インタープリタ

【図1】

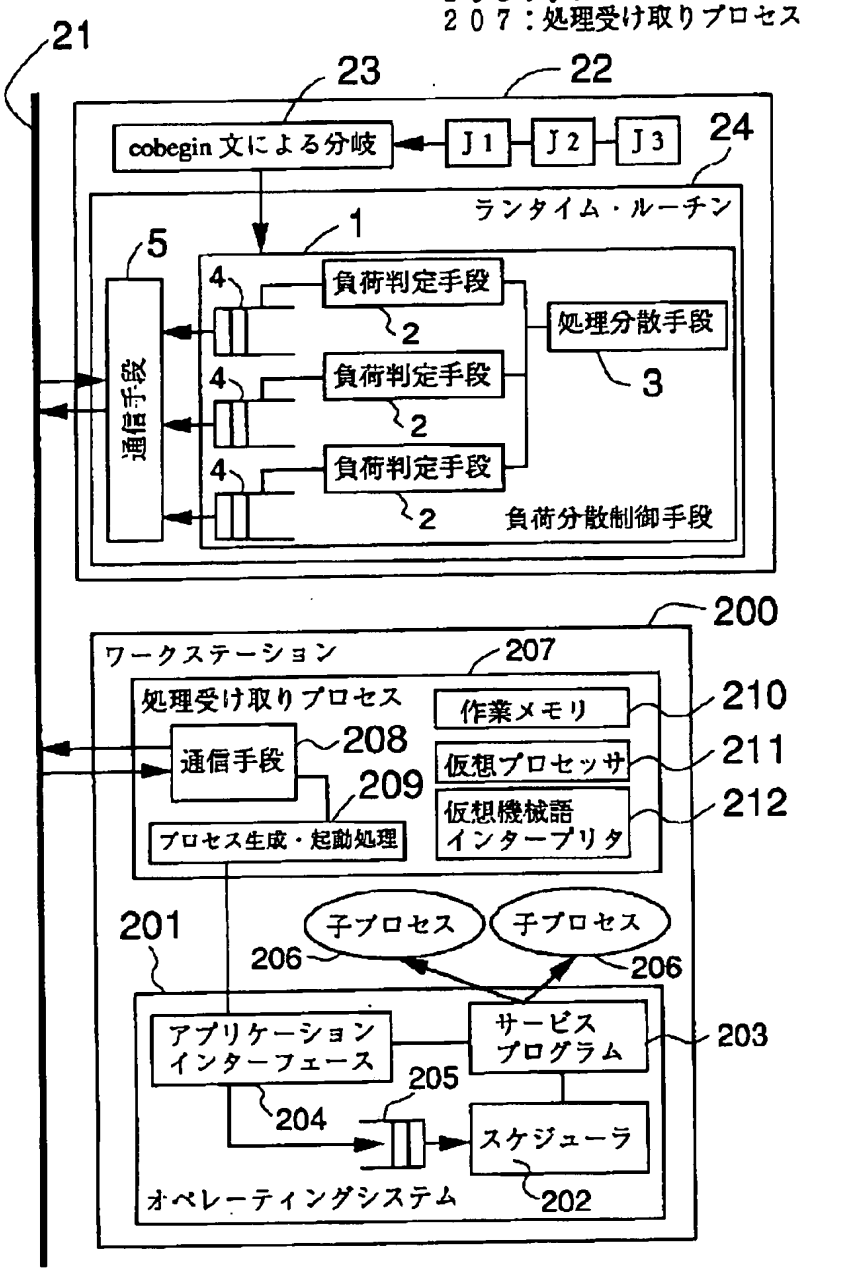


【図3】



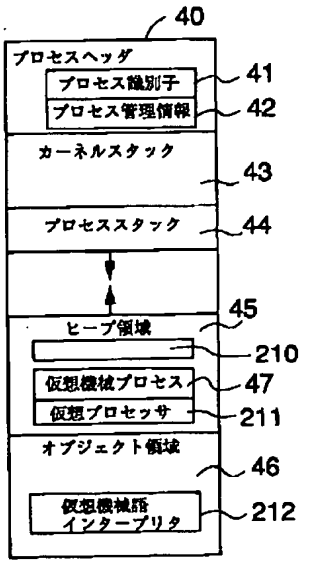
【図2】

２１：ネットワーク
２２：処理要求発生源プロセス
２００：ワークステーション
２０１：オペレーティングシステム
２０６：子プロセス
２０７：処理受け取りプロセス

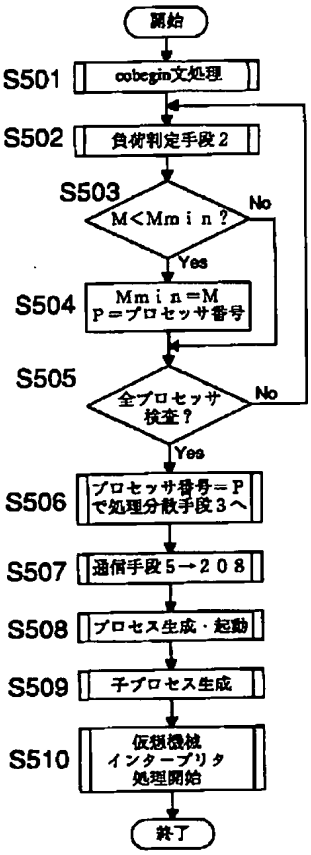


【図4】

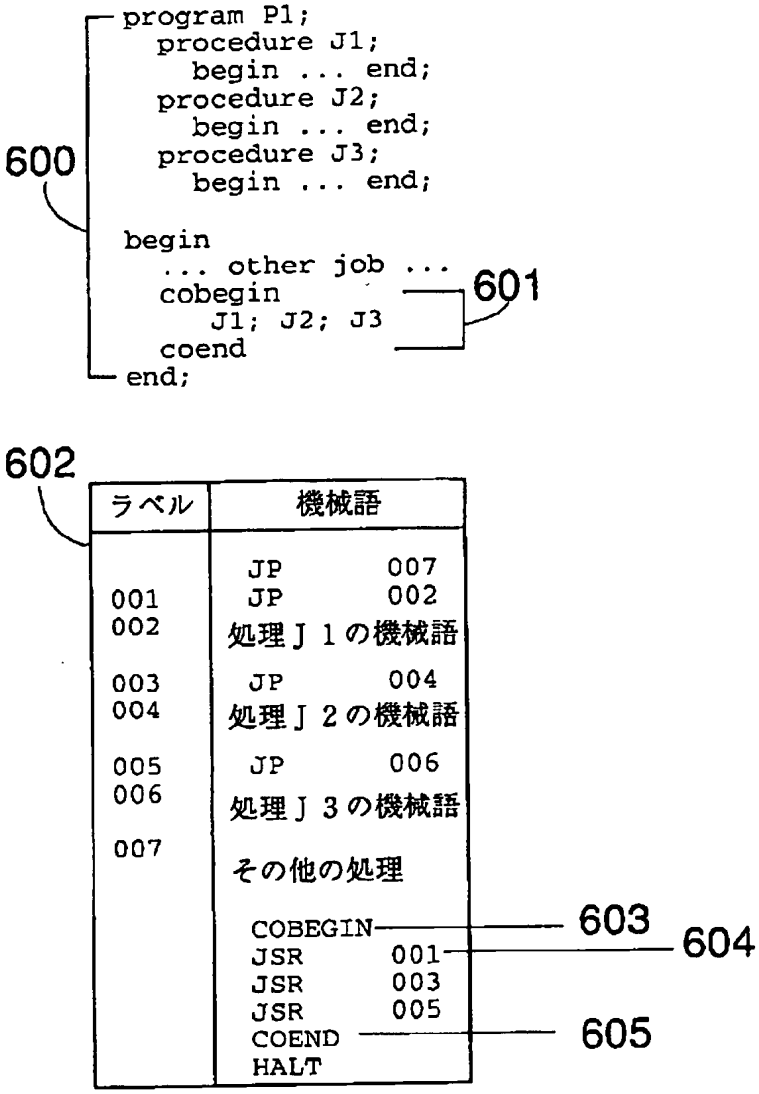
４０：プロセスの構造



【図5】



【図6】


program P1;
procedure J1;
begin ... end;
procedure J2;
begin ... end;
procedure J3;
begin ... end;
600
begin
... other job ...
cobegin
J1; J2; J3
601
coend
end;
602
ラベル
機械語
JP 007
001 JP 002
002 処理Ｊ１の機械語
003 JP 004
004 処理Ｊ２の機械語
005 JP 006
006 処理Ｊ３の機械語
007 その他の処理
COBEGIN
603
JSR 001
604
JSR 003
JSR 005
COEND
605
HALT